Edu Komputika 5 (1) (2018)

Edu Komputika Journal

http://journal.unnes.ac.id/sju/index.php/edukom

Evaluasi Metode *Load Balancing* dan *Fault Tolerance* pada Sistem *Database Server* Aplikasi *Chat*

Cakra Aminuddin Hamka ✉ , Teguh Bharata Adji dan Selo Sulistyo

Departemen Teknik Elektroda Teknologi Informasi, Universitas Gadjah Mada, Yogyakarta, Indonesia.

Info Artikel

Sejarah Artikel:

Diterima: 30 April 2018

Disetujui: 19 Juli 2018

Dipublikasikan: 20 Juli 2018

Keywords:
sistem, mesin, diagnosa, mobile.

Abstrak

Penelitian ini bertujuan untuk mengatasi besar dan banyaknya jumlah pengguna yang mengakses layanan aplikasi chat dan social. Kondisi tersebut dapat mengakibatkan komunikasi ke database server tunggal dapat terhenti dan hilangnya data dikarenakan adanya beban berlebihan. Metode yang digunakan dalam penelitian in adalah sistem database server lebih dari satu yang bertujuan meningkatkan ketersediaan database server dan pendistribusian layanan permintaan komunikasi yang menerapkan metode load balancing menggunakan HAPRoxy. Terdapat dua algoritma penjadwalan yang digunakan yaitu algoritma Round Robin dan Least Connection. Hasil dari evaluasi dan perbandingan kedua algoritma tersebut didapati nilai rata-rata algoritma least connection memiliki nilai waktu respon sebesar 32,421 ms lebih kecil dibandingkan dengan algoritma round robin yaitu 35,813 ms. Sedangkan pada throughput, algoritma least connection juga memiliki nilai yang lebih besar yaitu 211,267 Kb/s dibandingkan dengan algoritma round robin yang memiliki nilai 210,298 Kb/s. Hal ini menunjukkan jumlah paket pada algoritma least connection lebih besar dan waktu respon lebih cepat dari algoritma round robin. Hasil penelitian menunjukkan algoritma least connection lebih baik dibandingkan dengan algoritma round robin dan dengan implementasi load balancing pendistribusian komunikasi layanan pada database server lebih dari satu dapat mengatasi permintaan layanan komunikasi yang banyak dan besar.

Abstract

This study aims to address the large and large number of users accessing chat and social app services. Such conditions may result in communication to a single server database may be interrupted and data loss due to excessive load. The method used in this research is a database server system more than one that aims to improve the availability of database servers and distribution of demand for communication services that apply load balancing methods using HAPRoxy. There are two scheduling algorithm that used Round Robin and Least Connection algorithm. The result of evaluation and comparison of both algorithms found that the least connection algorithm has a response time value of 32,421 ms smaller than round robin algorithm that is 35,813 ms. While on throughput, least connection algorithm also has bigger value that is 211,267 Kb / s compared with round robin algorithm which have value 210,298 Kb / s. This shows the number of packets on the least connection connection algorithm and the faster response time of the round robin algorithm. The results show that least connection algorithms are better than round robin algorithms and with the implementation of load balancing the distribution of service communications on more than one server database can overcome the demand for large and large communication services.

✉ Alamat korespondensi:
Gedung E11 Lantai 2 FT Unnes
Kampus Sekaran, Gunungpati, Semarang, 50229
E-mail: cakra.mti15@mail.ugm.ac.id

ISSN 2252-6811

PENDAHULUAN

Salah satu tantangan utama komputasi penyimpanan data adalah meningkatnya jumlah data tiap waktu sehingga berdampak pada meluapnya jumlah data di media penyimpanan, seperti layanan teknologi *social media* (inet.detik.com, 2017). Salah satu aplikasi *social media* yang banyak di gunakan adalah berupa *chatting*. Pada umumnya untuk mengatasi masalah tersebut, kapasitas penyimpanan datanya ditambah. Beberapa penelitian ada yang menggunakan dan tetap mempertahankan sistem basis data tunggal (Hubspot, 2011). Akan tetapi sistem basis data tunggal memiliki kelemahan yaitu lambatnya komunikasi *big data* pada *server-database server*. Sehingga diperlukan adanya suatu sistem yang mampu mengelola komunikasi *big data* yang simultan dan stabil pada basis data (D. DeWitt, 1992). Pada penelitian sebelumnya salah satu solusi yang ditawarkan dari permasalahan tersebut adalah menggunakan metode menggunakan komunikasi terdistribusi berdasarkan perhitungan efisiensi dan penyediaan penyimpanan basis data (W.G. Aref, 2002) (Peter Mell, 2011). Proses pendistribusian berfungsi untuk membagi data, pemecahan dan jarak basis data untuk efisiensi komunikasi (Pakorn Kookarinat, 2015). Basis data terdistribusi didukung oleh operasi *throughput* yang menghitung jumlah kecepatan per detik pada proses penyimpanan sistem basis data (docs.oracle.com, 2017). Mekanisme pendistribusian basis data tersebut dapat bekerja secara optimal menggunakan alat tambahan yaitu *load balancing* (Xialin Wang, 2013). Akan tetapi metode pendistribusian menggunakan *load balancing* masih memiliki kelemahan yaitu sering terjadinya putus komunikasi, kehilangan data dan belum optimal dalam penyimpanan data tanpa adanya algoritma penjadwalan untuk pencarian jalur komunikasi basis data. Untuk mengatasi masalah terputusnya komunikasi, terdapat metode pendukung yaitu efisiensi komunikasi terdistribusi yang merata antara *server* ke *server* basis data menggunakan parameter *Quality of Service (QoS). QoS* adalah parameter untuk mengatasi masalah *fault tolerance* dan skalabilitas penyimpanan data yang konsisten (Tadeuz Pankowsi, 2010). *Fault tolerance* adalah sebuah sistem untuk memperbaiki kesalahan dan efisiensi komunikasi secara otomatis, serta mendistribusikan *bandwidth* data (F. Barse, 1973) (D. Mandelbaum, 1972) (L. Yang, 2001). Konsistensi dan efisiensi komunikasi dalam mendistribusikan *bandwidth* data bertujuan untuk penyimpanan data berskala besar dan respon cepat. Konsistensi tersebut dapat optimal menggunakan sistem komunikasi terdistribusi dan memaksimalkan nilai *throughput*. Nilai *throughput* adalah jumlah nilai rata-rata transfer *byte* yang ditransmisikan per satuan waktu (Catell R, 2010). Namun memiliki kelemahan tidak adanya penyeimbangan komunikasi data pada basis data.

Terdapat beberapa algoritma penjadwalan, yaitu seperti FIFO (First in First Out), SJF (Shortest Job First), Priority Scheduling, Round Robin dan least connection. Penelitian ini mengimplementasikan dua algoritma penjadwalan yang digunakan pada *load balancer* tersebut, yaitu algoritma *round robin* dan algoritma *least connection*. Proses alur algoritma Round Robin adalah menggilir proses antrian dalam rentang waktu. Kelemahan algoritma round robin adalah penentuan proses, apabila terlalu kecil, maka sebagian besar proses tidak akan selesai sesuai waktu. Kelebihan algoritma round robin adalah proses yang dilakukan adil, tidak ada prioritas dan tidak membutuhkan waktu lama. Sedangkan proses alur kerja algoritma Least Connection adalah melakukan pembagian beban berdasarkan banyaknya koneksi yang sedang dilayani. Koneksi yang paling sedikit akan diberikan beban koneksi berikutnya dan koneksi yang banyak akan di alihkan ke koneksi lain yang lebih rendah. Kelebihannya adalah perhitungan koneksi yang aktif untuk masing-masing koneksi dihitung secara dinamik dengan pendistribusian merata saat koneksi yang datang sangat banyak. Kelemahannya adalah terlalu banyaknya pengalihan koneksi pada sistem.

Beberapa penelitian telah menyelesaikan permasalahan sistem basis data tunggal dan pendistribusian komunikasi menggunakan *load balancer*. Tetapi penelitian tersebut memiliki kelemahan karena tidak adanya sistem

komunikasi terdistribusi *big data* simultan pada lebih dari satu *server* basis data. Kelemahan tersebut menyebabkan putus komunikasi dan hilangnya data dalam penyimpanan data pada *server* basis data. Untuk mengatasi masalah tersebut, penelitian ini mengusulkan suatu sistem yang mampu menangani terputusnya komunikasi data lebih dari satu *server* basis data menggunakan *load balancer*. *Load balancer* adalah sistem yang berfungsi untuk mengoptimalkan komunikasi pendistribusian beban kerja ke berbagai sumber daya komputasi menggunakan algoritma penjadwalan. Penelitian ini mengimplementasikan dua algoritma penjadwalan yang digunakan pada *load balancer* tersebut, yaitu algoritma *round robin* dan algoritma *least connection*. Sistem basis data terdistribusi yang menggunakan *load balancer* dapat diimplementasikan pada *mongodb, casandra, oracle* atau *MySQL*. Pada penelitian ini sistem pendistribusian data akan diimplementasikan menggunakan *mongodb*. Hal tersebut bertujuan untuk menjaga skalabilitas, stabilitas dan peningkatan komunikasi *big data* dan simultan pada lebih dari satu *server* basis data.

METODE PENELITIAN

LANDASAN TEORI

A. Chatting Jaringan Sosial

Jejaring sosial adalah layanan yang mampu membangun sistem publik oleh banyak pengguna perangkat komunikasi (E. Mit, 2012), tanpa harus saling bertatap muka (Obar, 2015). Interaksi komunikasi adalah metode komunikasi dua arah antara pengguna satu dan pengguna lainnya yang menggunakan media perangkat lunak (Haryadi, 2010). Terdapat beberapa contoh perangkat lunak antara lain aplikasi *chatting real-time* dan *multi-platform*.

B. Session

Session adalah pengiriman paket informasi yang menggunakan struktur pemilihan rute dengan parameter antrian data. Tujuan *session* adalah untuk memastikan toleransi kesalahan, pengiriman dan penyimpanan data komunikasi *real-time* (Stefano-Niko Orzen, 2017).

C. Quality of Service (QoS)

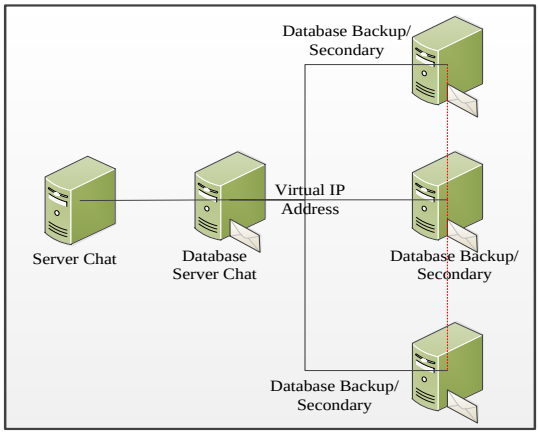
Quality of Service (QoS) adalah pengukuran performa layanan media, seperti jaringan data telepon atau komputer. Pengukuran kualitas layanan terdiri dari semua aspek, seperti layanan waktu respon, kehilangan koneksi, *echo*, *interrupts*, *frequency response* dan sebagainya (ITU-T, 2007).

D. *HAProxy*

HAProxy adalah salah satu perangkat lunak *open source*. *HAproxy* menyediakan ketersediaan *load balancer* yang tinggi, *proxy* untuk *server TCP* dan *HTTP* mendistribusikan permintaan komunikasi (Severalines AB, 2011). *HAProxy* dapat berjalan pada sistem operasi *Linux 2.4*, *Linux 2.6*, *Solaris 8/9*.

E. Arsitektur Basis Data Terdistribusi

Arsitektur yang baik diperlukan untuk mendukung layanan jaringan sosial dengan skalabilitas dan ketersediaan tinggi pada *server* basis data (Haproxy, 2018). Model empat (4) *server* basis data seperti ditunjukkan pada Gambar 1.



Gambar 1. Arsitektur *server* basis data Metode dan Parameter Pengukuran

F. Metode dan Parameter Pengukuran

1. Waktu Respon

   Waktu respon adaah total waktu pada saat penerimaan, antrian dan respon dari komunikasi data. Waktu respon dipengaruhi faktor kapasitas dan trafik komunikasi (Lopez, 2004). Perhitungan waktu respon menggunakan rumus Persamaan (2.1).

2. Throughput

   Throughput adalah jumlah rata-rata *byte* yang ditransmisikan per satuan waktu *bits per second (bps)* (27) (Ray S, 2012). Rata-rata jumlah *byte* pada *throughput* dipengaruhi oleh jumlah komunikasi pada rentang waktu tertentu (McBee J, 2010). Persamaan (2.2) merupakan rumus perhitungan nilai *throughput* (Brownlee N, 2001).
   Throughput = Jumlah Bit (kb) / Waktu Pengiriman (s) (2.2).

3. Fault *Tolerance*/ Toleransi Kesalahan

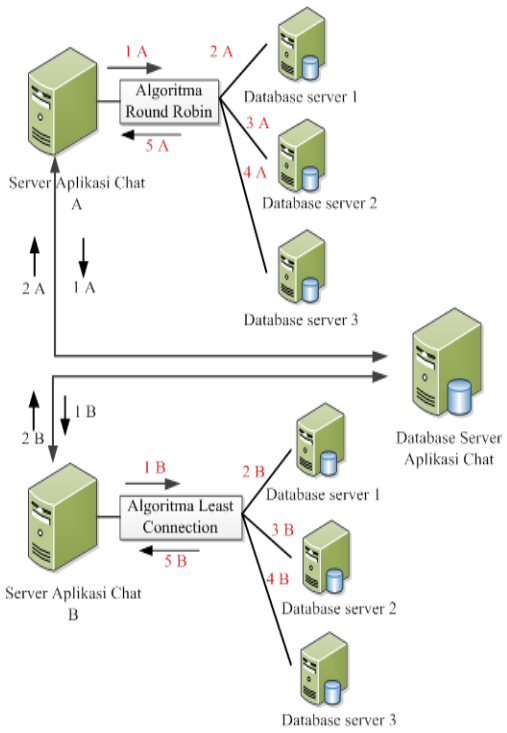
   Fault tolerance adalah kehandalan (reliability) dan ketersediaan (availability) komunikasi (Haproxy, 2018). Keandalan fault tolerance dipengaruhi oleh *Mean Time to Failure (MTTF)* dan *Mean Time Between Failure (MTBF)*. *MTTF* merupakan waktu rata-rata sistem beroperasi sampai terjadi kegagalan. *MTFB* adalah waktu rata-rata antara dua kegagalan yang terjadi berturut-turut. *Mean Time to Repair (MTTR)* merupakan perbedaan antara MTTF dan MTBF adalah jumlah waktu untuk memperbaiki sistem setelah terjadi kegagalan sistem (McBee J, 2010) (Alimuddin, 2011). Seperti ditunjukkan Persamaan (2.3) dan (2.4).
   MTBF (detik) = MTTF (detik) + MTTR (detik) (2.3)
   Availability = MTTF (detik) /MTBF (detik) = MTTF (detik) / (MTTF (detik)+MTTR (detik)) (2.4).

SISTEM DAN ALGORITMA

Pada penelitian ini mengusulkan sistem basis data terdistribusi menggunakan *load balancer* untuk menjaga stabilitas komunikasi antara *server chatting* dan *server* basis data lebih dari satu. Penelitian ini mengimplementasikan pada setiap *server* aplikasi *chat* adalah 3 (tiga) buah *server* basis data. *Server* basis data 1 adalah sebagai media penyimpanan dan komunikasi utama antara *server* aplikasi *chat* dengan sistem *server* basis data. *Server* basis data 2 adalah sebagai media penyimpanan dan komunikasi cadangan apabila *server* basis data 1 tidak aktif/*down*. *Server* basis data 3 adalah sebagai cadangan sistem penyimpanan, komunikasi, sistem pendistribusian dan replikasi basis data. *Load balancing* sistem terdiri dari algoritma penjadwalan *round robin* dan algoritma *least connection* dijelaskan pada sesi ini:

A. Sistem *Server* Basis Data *Load Balancing*

Tujuan sistem *load balancing* adalah untuk mengoptimalkan beban kerja distribusi komunikasi ke berbagai sumber daya komputasi, memaksimalkan *throughput* dan meminimalkan waktu respon. *Load balancing* berfungsi untuk pengalihan lalu lintas komunikasi data saat terjadi *big data* dan simultan. Jika lonjakan lalu lintas komunikasi data pada *server* basis data dibiarkan, maka dapat menyebabkan putusnya komunikasi data ke *server* basis data. Load balancer pada server dibangun untuk mengatasi tersebut, hal ini untuk menyimpan data komunikasi ke *server* basis data lebih dari satu. Skema sistem *server* basis data dan alur layanan komunikasi ditunjukkan seperti Gambar 2.



Gambar 2. Sistem *Server* Basis data dan alur komunikasinya.

Sistem load balancer sebagai sebuah solusi digunakan untuk mendistribusikan komunikasi *big data*. Sistem ini terdiri dari satu buah *server*, satu buah *load balancing* dan tiga *server* basis data. Strategi skema tersebut dapat mendukung peningkatan lalu lintas komunikasi *server* ke *server* basis data *chatting social network*.

Fungsi lain dari *load balancing* adalah menghitung algoritma penjadwalan *least connection* dan *round robin*. Nilai rata-rata kedua algoritma tersebut dihitung nilai waktu respon yang kecil dan nilai jumlah paket data pada *throughput* yang besar. Skema perhitungan untuk mencari nilai waktu respon dan nilai *throughput* dapat dilakukan dengan beberapa kondisi, diantaranya adalah:

1. Kondisi *load balancer* aktif dan *server* basis data *mongodb standby*.
2. Kondisi *load balancer* dinonaktifkan dan komunikasi ke *server* basis data *mongodb standby*.
3. Kondisi *load balancer* diaktifkan kembali, dan komunikasi normal kembali.

B. *Algoritma Round Robin*

Algoritma *Round Robin* adalah proses *time sharing* yang membagi beban komunikasi data pada *load balancer* secara bergiliran dan berurutan dari satu *server* ke *server* lainnya (Obar, 2015). Proses pembagian data tersebut berbasis node penyeimbang beban komunikasi ke server (Haryadi, 2010). *Node* adalah sebuah bobot dari nilai integer yang menunjukkan kapasitas pemrosesan dalam urutan periode tertentu (D. Henriyan, 2016). Alur proses algoritma *round robin* adalah dengan metode (1/n) dan waktu (n-1) q.

C. *Algoritma Least Connection*

Algoritma *least connection* adalah algoritma menghitung koneksi layanan paling sedikit untuk setiap *node*nya dan pendistribusian ke semua *server*nya (Stefano-Niko Orzen, 2017).

HASIL DAN PEMBAHASAN

Tujuan dari metode *load balancing* pada *server* basis data sebagai metode yang diusulkan adalah untuk mengetahui performa dari pendistribusian komunikasi dan penyimpanan data pada *server* basis data. Terdapat dua parameter yang diukur untuk melihat performa dan mengevaluasi kinerja *load balancing* dalam mendistribusikan layanan komunikasi. Dua parameter tersebut adalah nilai waktu respon dan throughput digunakan sebagai acuan dalam menilai QoS dari sistem server basis data untuk aplikasi chatting social network. Httperf adalah perangkat lunak untuk mendukung pengujian metode *load balancing*.

1. Alamat IP pada *server* basis data dan *load balancer*

Pengaturan alamat IP pada setiap *server* basis data dan *load balancer* harus dilakukan untuk memudahkan komunikasi antara *server* basis data dan *load balancer*. Pengaturan alamat IP ini ditunjukkan pada Tabel 1:

Tabel 1. Nama *server* basis data dan alamat IP

No	Nama	Alamat IP
1	*Load Balancer* *Server* Basis Data	192.168.168.131
2	*Server* Basis Data 1	192.168.34.23
3	*Server* Basis Data 2	192.168.34.24
4	*Server* Basis Data 3	192.168.34.25
5	*Server* Basis Data 4	192.168.34.26
6	*Server* Basis Data 5	192.168.34.27
7	*Server* Basis Data 6	192.168.34.28

2. Uji Algoritma *Round Robin* dan Algoritma *Least Connection*

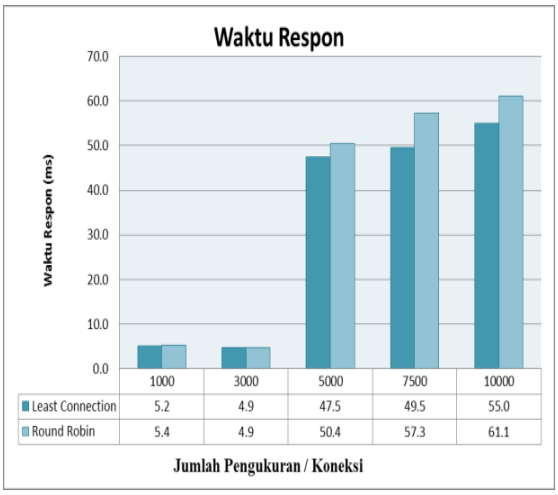
Pengujian kedua algoritma ini bertujuan untuk mengetahui performa pendistribusian dan komunikasi data. Pengujian dilakukan dengan membuat permintaan layanan koneksi dari *server chatting* ke *server* basis data sebanyak 50 kali. Hasil dari pengujian ini lalu diamati dan dibandingkan pada kedua algoritma penjadwalan yang diimplementasikan. Pengujian jumlah koneksi dapat dijelaskan pada Tabel 2.

Tabel 2. Jumlah Koneksi pada *Load Balancing.*

No	Jumlah Koneksi	Permintaan / Detik	Banyak Pengujian
1	1000	100	50
2	3000	300	50
3	5000	500	50
4	7500	750	50
5	10000	1000	50

3. Uji Waktu Respon pada kedua Algoritma

Hasil rata-rata pengukuran waktu respon dari permintaan komunikasi ke *server* basis data *load balancing*, algoritma *least connection* dan algoritma *round robin* kemudian di bandingkan. Hasil akan dtunjukkan grafik pada Gambar 3.

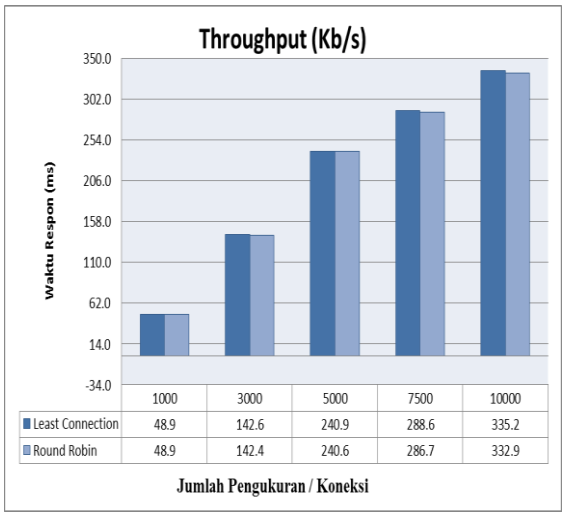


Gambar 3. Grafik Hasil Pengukuran Waktu Respon.

Gambar 3 menunjukkan adanya perbedaan data antara algoritma *Round Robin* dan *Least Connection* dari uji parameter waktu respon. Pada komunikasi sebesar 1000 koneksi adanya perbedaan 0,2. Komunikasi sebesar 3000 tidak adanya perbedaan, saat koneksi sebesar 5000 perbedaan data sebanyak 3.0. Komunikasi yang besar pada 7500 koneksi perbedaan semakin besar hingga 7.8 dan koneksi sebesar 10.000 perbedaan data sebesar 6.0.

4. Uji *throughput*

Hasil rata-rata pengukuran *throughput* permintaan komunikasi ke *server* basis data *load balancing* dan pengujian algoritma *least connection* dan algoritma *round robin* kemudian di bandingkan. Hasil yang diperoleh berupa data satuan *kilobyte per second (KB/s)* pada grafik Gambar 4.



Gambar 4 Grafik Hasil Pengukuran *Throughput.*

Berdasarkan Gambar 4 memperlihatkan adanya perbedaan data antara algoritma *Round Robin* dan *Least Connection* dari uji parameter *throughput*. Pada komunikasi sebesar 1000 koneksi tidak ada perbedaan. Komunikasi sebesar 3000 terdapat perbedaan 0.2 kb/s, koneksi sebesar 5000 perbedaan data sebanyak 0.3. Komunikasi yang besar pada 7500 koneksi perbedaan semakin besar hingga 1.9 kb/s dan koneksi sebesar 10.000 perbedaan data sebesar 2.4 kb/s.

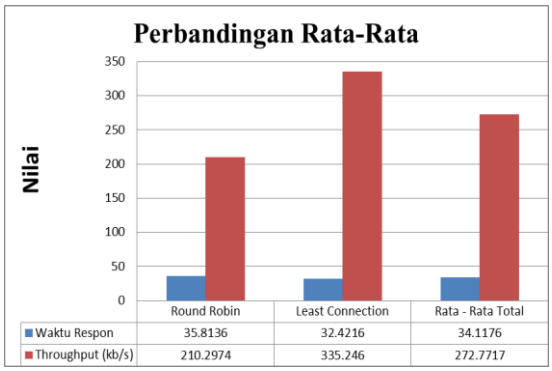
Hasil dari perbandingan waktu respon dan nilai *throughput* pada kedua algoritma *round robin* dan *least connection* akan disajikan pada Tabel 5.

Tabel 3. Hasil permintaan layanan pada algoritma *round robin* dan *least connection*

Jumlah Permintaan	Round Robin		Least Connection	
	Waktu respon (ms)	Throughput (kb/s)	Waktu respon (ms)	Throughput (kb/s)
100 req/s 1000 koneksi	5,388	48,9	5,188	48,926
300 req/s 3000 koneksi	4,888	142,4	4,888	142,648
500 req/s 5000 koneksi	50,444	240,604	47,484	240,878
750 req/s 7500 koneksi	57,274	186,729	49,514	288,6431
1000 req/s 10000 koneksi	61,074	332,854	55,034	335,246
Rata-rata	**35,8136**	**210,2974**	**32,4216**	**211,26822**

5. Perbandingan Pengujian waktu respon dan *throughput*

Berdasarkan dari himpunan data dari Tabel 3 waktu respon dan nilai *throughput* dari algoritma *least connection dan round robin*, maka diambil sebuah rata-rata perbandingannya, seperti yang terlihat pada grafik Gambar 5.



Gambar 5. Perbandingan Rata-Rata waktu respon dan *throughput* pada algoritma least connection dan round robin.

6. Pengujian Sebelum Penerapan Sistem *Load Balancing*

Pengujian ini dilakukan untuk mengetahui perbandingan skalabilitas sebelum penerapan *load balancing* yaitu dengan menggunakan basis data tunggal. Pengujian ini dilakukan dengan masing-masing parameter, yaitu waktu respon dan nilai *throughput*. Dari hasil pengujian tersebut disajikan pada Tabel 4.

Tabel 4 Pengujian Sebelum Penerapan *Load Balancing*.

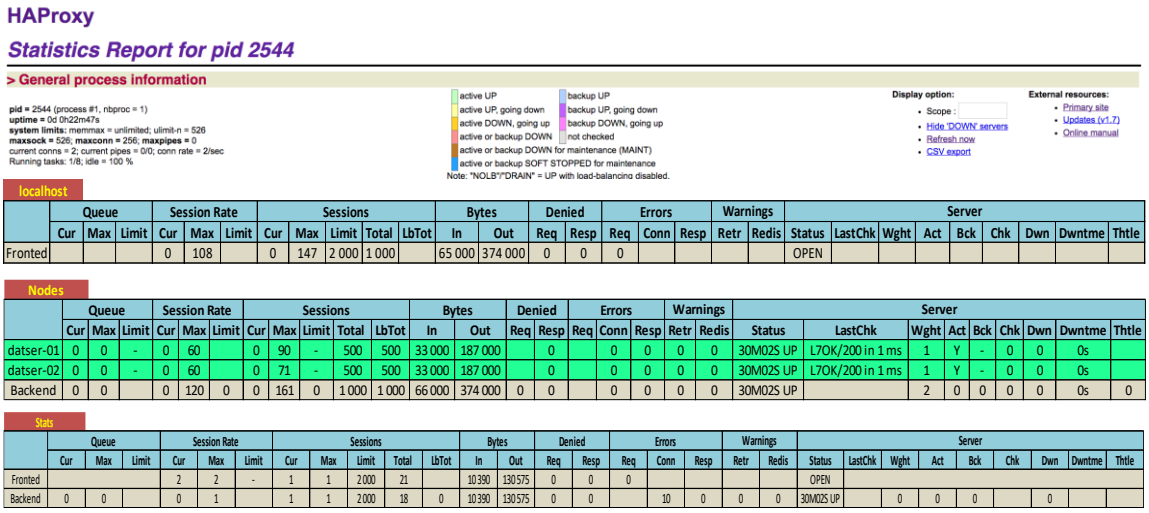
Jumlah Pengujian	**Sebelum Penerapan Load Balancer**	
	Waktu Respon	**Throughput**
1000	7.0662	48.9260
3000	1061.8816	120.130
5000	0.0	0.0
7500	0.0	0.0
10000	0.0	0.0
Rata-Rata	**213.7896**	**33.8112**

Dari hasil pengujian pada basis data tunggal ini menunjukkan nilai yang cukup tinggi pada pengujian ke 1000 dan 3000. Namun pada pengujian koneksi pada 5000 hingga 10000 koneksi, basis data tunggal sudah tidak mampu menangani proses komunikasi tersebut.

7. Pengujian Tiga Kondisi *Server* Basis Data

A. Kondisi *Server* Basis data *Load Balancer* Aktif.

Pada pengujian pertama ini *server* basis data *load balancer* dalam keadaan aktif dan semua *server* basis data dalam kondisi aktif pasif. Kemudian dilakukan pengaksesan sistem ke *server* basis data, disaat yang sama dilakukan pemantauan *server* secara *real time*. Hasilnya adalah semua berjalan dengan normal 99.99%. Hasil ini terlihat pada *haproxy* pada Gambar 6.



HAProxy

Statistics Report for pid 2544

> General process information

pid = 2544 (process #1, nbproc = 1)
uptime = 0d 0h22m47s
system limits: memmax = unlimited; ulimit-n = 526
maxsock = 526; maxconn = 256; maxpipes = 0
current conns = 2; current pipes = 0/0; conn rate = 2/sec
Running tasks: 1/8; idle = 100 %

active UP
active UP, going down
active DOWN, going up
active or backup DOWN
active or backup DOWN for maintenance (MAINT)
active or backup SOFT STOPPED for maintenance
backup UP
backup UP, going down
backup DOWN, going up
not checked
Note: "NOLB"/"DRAIN" = UP with load-balancing disabled.

Display option:
- Scope :
- Hide 'DOWN' servers
- Refresh now
- CSV export

External resources:
- Primary site
- Updates (v1.7)
- Online manual

localhost

	Queue			Session Rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thtle
Fronted				0	108		0	147	2 000	1 000		65 000	374 000	0	0	0					OPEN								

Nodes

	Queue			Session Rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thtle
datser-01	0	0	-	0	60		0	90	-	500	500	33 000	187 000		0		0	0	0	0	30M02S UP	L7OK/200 in 1 ms	1	Y	-	0	0	0s	
datser-02	0	0	-	0	60		0	71	-	500	500	33 000	187 000		0		0	0	0	0	30M02S UP	L7OK/200 in 1 ms	1	Y	-	0	0	0s	
Backend	0	0		0	120	0	0	161	0	1 000	1 000	66 000	374 000	0	0		0	0	0	0	30M02S UP		2	0	0	0	0	0s	0

Stats

	Queue			Session Rate			Sessions					Bytes		Denied		Errors			Warnings		Server								
	Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	In	Out	Req	Resp	Req	Conn	Resp	Retr	Redis	Status	LastChk	Wght	Act	Bck	Chk	Dwn	Dwntme	Thtle
Fronted				2	2	-	1	1	2 000	21		10 390	130 575	0	0	0					OPEN								
Backend	0	0		0	1		1	1	2 000	18	0	10 390	130 575	0	0		10	0	0	0	30M02S UP		0	0	0		0		

Gambar 6 Pemantauan Komunikasi Kondisi Server Basis Data *Load Balancer* Aktif.

B. Kondisi *Server* Basis Data *Load Balancer* Di Nonaktifkan.

Pengujian dilakukan untuk mengetahui berapa lama perpindahan komunikasi dari *server* basis data 1 ke *server* basis data 2 yang mengimplementasikan *load balancer*. Skema pengujian tersebut dihasilkan kegagalan koneksi selama 215 detik, dan untuk pemulihan sistem selama 2 detik. Dari hasil pengujian tersebut, kemudian di hitung persentase ketersediaan menggunakan persamaan (2.4) pada sistem *server* basis data berdasarkan *fault tolerance* yaitu:

MTBF = MTTR/ (MTTR MTTR)

$$\frac{215}{215+2} = 0.9907\ atau\ 99.07\%$$

Berdasarkan perhitungan *fault tolerance*, maka didapati hasil sebesar 99,07% ketersediaan koneksi layanan permintaan pada *server* basis data saat dinonaktifkan.

C. Kondisi *Server* Basis Data *Load Balancer* Di Aktifkan

Pengujian dilakukan untuk mengetahui berapa lama perpindahan komunikasi dari *server* basis data 2 ke *server* basis data 1 yang mengimplementasikan *load balancer*. Skema pengujian tersebut dihasilkan kegagalan koneksi selama 60 detik, dan untuk pemulihan sistem selama 2 detik. Dari hasil pengujian tersebut, kemudian dihitung persentase ketersediaan sistem *server* basis data berdasarkan *fault tolerance* yaitu:

MTBF = MTTR/ (MTTR MTTR)

$$= \frac{60}{60+2} = 0.9677\ atau\ 97\%$$

Berdasarkan perhitungan *fault tolerance* saat kondisi *server* basis data *load balancer* di aktifkan kembali, maka didapati hasil sebesar 97% ketersediaan koneksi layanan permintaan pada *server* basis data saat di aktifkan.

KESIMPULAN

Dari hasil penelitian, pengujian, analisis data dan pembahasan, dapat disimpulkan bahwa: Algoritma least connection memiliki kinerja lebih baik dari algoritma round robin, hal ini ditunjukkan dengan nilai rata-rata pengujian koneksi least connection memiliki waktu respon 32,4216 ms lebih kecil dibandingkan round robin 35,8136 ms. Kemudian *throughput*, algoritma least connection memiliki nilai lebih besar yaitu 211,26822 kb/s dibandingkan dengan round robin 210,2974 kb/s. Metode *load balancing* layak diimplementasikan pada sistem *server* basis data lebih dari 1, karena mampu melayani hingga 10000 koneksi dengan 1000 request/detik, sedangkan sistem tanpa *load balancing* dan sistem *server* basis data tunggal hanya mampu melayani 3000 koneksi dengan 300 request/detik. *Server* basis data *load balancer* telah mampu mempertahankan permintaan dan ketersediaan *server* basis data. Terbukti ketika *server* basis data *load balancer* utama mengalami masalah, maka secara otomatis perannya digantikan dengan *server* basis data *load balancer* kedua dalam kurun waktu hanya 2 detik untuk pemulihan sistem.

SARAN

1. Perlu adanya pengujian lebih lanjut mengenai algoritma penjadwalan lain dengan metode *load balancing*.
2. Perlu adanya penelitian lebih lanjut tentang *server* basis data 3 yang dapat menjadi sistem restore pada *server* basis data 1 dan *server* basis data 2.
3. Perlu adanya penelitian lebih lanjut mengenai sebuah sistem keamanan untuk keseluruhan sistem *server* basis data dan sistem *chatting social media.*

DAFTAR PUSTAKA

Alimuddin, dan Ashari, A. 2011. "Peningkatan Kinerja SIAKAD Menggunakan Metode Load balancing dan Fault Tolerance Di Jaringan Kampus Universitas Halu Oleo," Jurnal IJCCS, Vol.10 No.1.

Brownlee N, dan Loosley, C. 2001. Fundamental of Internet Measu rement: A Tutorial. CGM Journal of Computer Resource Management 102.

Cattell, R. 2010. Scalable SQL and NoSQL data stores. SIGMOD Record, 39(4):12–27.

D. DeWitt and 1. Gray. 1992, Juni. Parallel database systems: the future of high performance database systems. Commun. ACM, vol. 35, no. 6, pp. 85-98.

D. Henriyan, D. P. Subiyanti, R. Fauzian, D. Anggraini, and M. V. G. Aziz. 2016. "Design and Implementation of Web Based Real Time Chat Interfacing server," pp. 83-87.

D. Mandelbaum. 1972, June. "Error correction in residue arithmetic", IEEE Transactions on Computers, Volume 21, No. 6, pp 538-545.

E. Mit, N. H. Borhan, and M. A. Khairuddin. 2012. "Need Analysis of Culture-based Genealogy Software for Indigenous Communities," 2012 IEEE Symp. E-Learning, E-Management, E-Services, IS3e 2012, pp. 61-65.

Ferdinando. 2004. Fault Tolerance in Real-time Distributed System Using the CT Library. Master's Thesis.Department of Electrical Engineering, Faculty EE-Math-CS. University of Twente. Belanda.

F. Barse and P. Meastrini. 1973, March. "Error correction properties of redundant residue number systems", IEEE Transactions on Computers, Volume 22, No. 3, pp 307-315.https://inet.detik.com/cyberlife/d-3659956/132-juta-pengguna-internet-indonesia-40-penggila-medsos.

Haproxy. 2018, November. Retrieved from https://www.haproxy.org.

https://www.hubspot.com/hs-fs/hub/53/file-13206174-pdf/docs/marketingcharts-social-media-data-stacks-pdf.pdf

https://docs.oracle.com/cd/E17276_01/html/programmer_reference/transapp_throughput.html

Lopez, T.S., 2004. Analisys on Linux Server Clustering. Thesis. Faculty of Computer Sience. Polytechnic University of Valencia.

L. Yang and L. Hano. 2001, April. "Redundant Residue Number System Based Error Correction Codes", IEE Electronics Letters, Volume 37, No. 4, pp 247-248.

Madalina, M., dan Bucharest. 2007. Analyzing the Network Response Time and Load balancing. Journal Revista Informatica Economica, nr.

McBee, J., dan Elfassy, D., 2010. Mastering Microsoft Exchange Server 2010. Wiley Publishing, Inc., ISBN: 978-0-470-52171-7. Indianapolis, Indiana.

"MySQL Load Balancing with HAProxy". Severalnines AB. 2011. Retrieved 19 February 2013.

Obar, Jonathan A., Wildman, Steve. 2015. "Social media definition and the governance challenge: An introduction to the special issue". Telecommunications policy. 39 (9): 745–750.

Pakorn Kookarinrat, Yaowadee Temtanapat. 2015. "Analysis of range-based key properties for sharded cluster of MongoDB", IEEE 978-1-4673-8611-1/15.

Peter Mell, Tim Grance. 2011, 24 July. "The NIST Definition of Cloud Computing". National Institute of Science and Technology.

Ray, S dan Sarkar, A.D. 2012. Execution Analysis of load balancing Algorithms in Cloud Computing Environment. International. Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.5. Department of Computer Science and Engineering, Birla Institute of Technology, Mesra, Kolkata.

Stefano-Niko Orzen, Sorin Babii. 2017. "Network Events in the Dynamic Selection of Real-Time Session Fault Tolerant Routes". INES – IEEE.

Tadeusz Pankowsi, "Consistency and Availability of Data in Replicated NoSQL Databases", Institute of Control and Information Engineering, Poznan University of Technology, Poznan, Poland.

"Teletraffic Engineering Handbook". 2007, January 11. at the Wayback Machine. ITU-T Study Group 2 (350 pages, 4·48MiB) (It uses abbreviation GoS instead of QoS).

W.G. Aref, A.C. Catlin, A.K. Elmagarmid, J. Fan and Friends. 2002. "A Distributed Database Server for Continuous Media", Department of Computer Sciences, Purdue University, West Lafayette IN 47907-1398, Proceedings of the 18th International Conference on Data Engineering, 1063-6382/02 IEEE.

Xiaolin Wang, Haopeng Chen and Zhenhua Wang. 2003. "Research in Improvement of Dynamic Load Balancing in MongoDB", IEEE 11th International Conference on Dependable, Aoutonomic and Secure Computing, 978-1-4799-3381-5/13, DOI 10.1109/DASC.2013.49.